Web-basierte Geoprozessierung mit Python und PyWPS

Jonas Eberle
Friedrich Schiller Universität Jena
Institut für Geographie
Lehrstuhl für Fernerkundung

Email: jonas.eberle@uni-jena.de
Web: www.eo.uni-jena.de

Über mich

Jonas Eberle

- Wissenschaftlicher Mitarbeiter, Doktorand an der Universität Jena
- Freiberuflicher Web-und Software-Entwickler

Ausbildung

- Bachelor of Science Angewandte Informatik mit Studienrichtung Umweltinformatik, Hochschule Ostwestfalen
- Master of Science Geoinformatik & Fernerkundung, Universität Jena

Mein Fokus

- Web-basierte Tools für den Zugriff und die Analyse von Erdbeobachtungs-Zeitreihendaten basierend auf OGC-kompatiblen Diensten für Web- und Mobil-Anwendungen

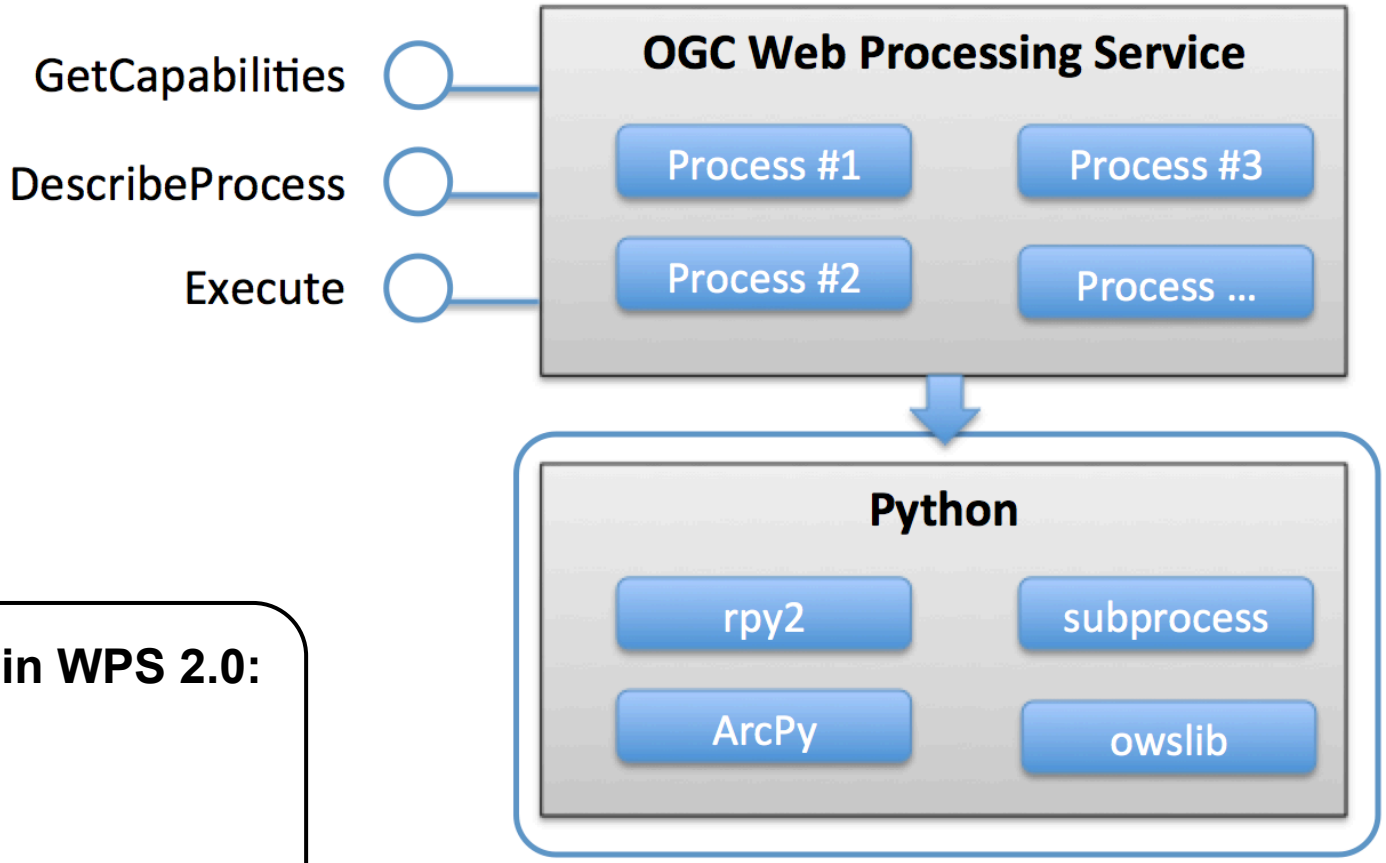
Motivation

- Warum brauchen wir web-basierte Geoprozessierungsdienste?
  - Verarbeitung von Geodaten über das Internet
    - z.B. Transformation von Koordinaten, Berechnung des Höhenprofils
    - Zugriff auf externe Datenprovider inklusive der Vorverarbeitung der Daten
  - Verstecken der Verarbeitungskomplexität
    - Parallele Verarbeitung durch Cloud / Cluster
    - Multi-core Verarbeitung
    - Abbauen von Barrieren in Datenzugriff und –analyse
  - Automatische Verknüpfung von Datenzugriff und -analyse
  - “Pay per use” business model?

- Warum sollte Geoprozessierungsdienste über WPS verfügbar sein?
  - Standard-kompatible Clients (z.B., QGIS Plugin, Python-Bibliothek “OWSlib”) können direkt mit den bereitgestellten Diensten arbeiten
  - Wiederverwendung von existierenden Bibliotheken und Anwendungen (z.B., Clients, Monitoring, Logging)

OGC Web Processing Service (WPS)

WPS Version 1.0



Neue Methoden in WPS 2.0:

- GetStatus
- GetResult
- Dismiss

WPS Beispielaufrufe

http://artemis.geogr.uni-jena.de/cgi-bin/testbox.cgi?service=WPS&version=1.0.0

request=**GetCapabilities**

request=**DescribeProcess**&identifier=1013_single_ts_plot_point

request=**Execute**&identifier=1013_single_ts_plot_point&datainputs=
[datasetName=mod13q1_evi;pointX=13.54;pointY=52.31]
&status=true&storeExecuteResponse=true *(optional)*

WPS Software (open source)

Software	Language	Supported process languages	Website
52° North WPS	Java	Java, R (WPS4R interface)	52north.org/wps
Degree WPS	Java	Java	www.deegree.org
ZOO WPS	C	C, Python, PHP, Java, JavaScript	www.zoo-project.org
PyWPS	Python	Python	pywps.wald.intevation.org
Geoserver WPS	Java	Java, Python	geoserver.org

WPS Prozessbeschreibung

Input- und Outputtypen

- **ComplexData**
  - i.d.R. genutzt für Raster- und Vektordaten
  - Referenziert als URL oder kodiert in Anfrage/Antwort
  - Datenformate: Raster, Vektor, XML, JSON, etc.

- **LiteralData**
  - genutzt für “einfache” Werte
  - Datenformate: Zeichenketten, Ganzahlen, Fließkommazahlen, etc.

- **BoundingBoxData**
  - Koordinatenpaare der Ecken in einem definierten Koordinatensystem
  - Format: minx, miny, maxx, maxy

Python

- Open Source Programmiersprache
- http://python.org

- **Standardbibliotheken**
  - os, sys
  - csv
  - json
  - requests (HTTP)

- **Geo/GIS-Bibliotheken**
  - Lesen, Schreiben, Transformieren: GDAL/OGR, Fiona, RasterIO
  - Geometrien: Shapely
  - Koordinaten: PyProj
  - Rasterverarbeitung: NumPy
  - GIS: PyGRASS, PyQGIS

- **Anwendungen**
  - Desktop-GIS Scripting (QGIS, GRASS GIS, ArcGIS)
  - WebGIS Server (pycsw, PyWPS)
  - WebGIS CMS (GeoNode, Django)
  - WebGIS Clients (OWSlib)

PyWPS

- Implementierung der OGC Web Processing Service (WPS) Spezifikation des OGC
- PyWPS ist geschrieben in Python
- Erste Entwicklung 2006 durch Jachym Cepicky
- Unterstützt alle verfügbaren Tools in Python
- http://pywps.org

- Aktuelle stabile Version: 3.2.6
- Komplette Neuentwicklung: PyWPS 4
- OSGeo Incubation dieses Jahr gestartet
- Google Summer of Code Projekte

PyWPS Workshop auf der FOSS4G:

- Online-Tutorials für PyWPS 3 und 4, siehe http://pywps.org

PyWPS 3 Prozessbeispiel #1

```
from pywps.Process import WPSProcess
import types
class Process(WPSProcess):
    def __init__(self):
        # init process
        WPSProcess.__init__(self,
            identifier = "test_prozess",
            title="Test Prozess",
            version = "0.1",
            storeSupported = "true",
            statusSupported = "true",
            abstract="liest einen LiteralInput vom Typ
                string ein und gibt ihn wieder aus",
            grassLocation =False)

        #Definition der Prozess Inputs
        self.str_in = self.addLiteralInput(
                        identifier = "str_in",
                        title = "String Input",
                        type=type('string'),
                        default='Test String',
                        )

        #Definition der Prozessoutputs
        self.string_output=self.addLiteralOutput(
                        identifier="str_out",
                        title="string output",
                        type=type('string'))

    def execute(self):
        self.str_out.setValue(self.str_in.getValue())
        return
```

Prozesseigenschaften

Prozess-Input(s)

Prozess-Output(s)

Prozessausführung

PyWPS 3 Prozessbeispiel #2

```
# process inputs
self.wkt = self.addLiteralInput(identifier="wkt",
    title = "wkt",
    abstract="",
    type=types.StringType,
    minOccurs=1,
    maxOccurs=1
)

self.epsg_source = self.addLiteralInput(identifier="epsg_source",
    title = "Source EPSG projection",
    abstract="",
    type=types.IntType,
    minOccurs=1,
    maxOccurs=1
)

self.epsg_target = self.addLiteralInput(identifier="epsg_target",
    title = "Target EPSG projection",
    abstract="",
    type=types.IntType,
    minOccurs=1,
    maxOccurs=1
)
```

```
# process output
self.output = self.addLiteralOutput(identifier="output",
    title = "WKT output",
    abstract="",
    type=types.StringType
)
```

PyWPS 3 Prozessbeispiel #2

```
def execute(self):

    # get input values
    wkt = self.wkt.getValue()
    epsg_source = int(self.epsg_source.getValue())
    epsg_target = int(self.epsg_target.getValue())

    osr.UseExceptions()

    # Achtung: Zur Uebersicht werden in diesem Beispiel keine Exceptions aufgefangen!
    source = osr.SpatialReference()
    source.ImportFromEPSG(epsg_source)

    target = osr.SpatialReference()
    target.ImportFromEPSG(epsg_target)

    geom = ogr.CreateGeometryFromWkt(wkt)
    transform = osr.CoordinateTransformation(source, target)
    geom.Transform(transform)

    output_wkt = geom.ExportToWkt()

    # return output
    self.output.setValue(output_wkt)
```

Client-Bibliotheken

- OpenLayers 2 WPS Client
  - http://dev.openlayers.org/examples/wps-client.html
- OpenLayers 3 WPS Client
  - https://github.com/boundlessgeo/wps-gui/blob/master/src/wpsclient.js
- Python „OWSlib“
  - https://geopython.github.io/OWSLib/#wps
- QGIS Desktop
  - https://plugins.qgis.org/plugins/wps/

Use case: Earth Observation Monitor

Datenzugriff und -analyse für Vegetationszeitreihen

Verwendete Datensätze

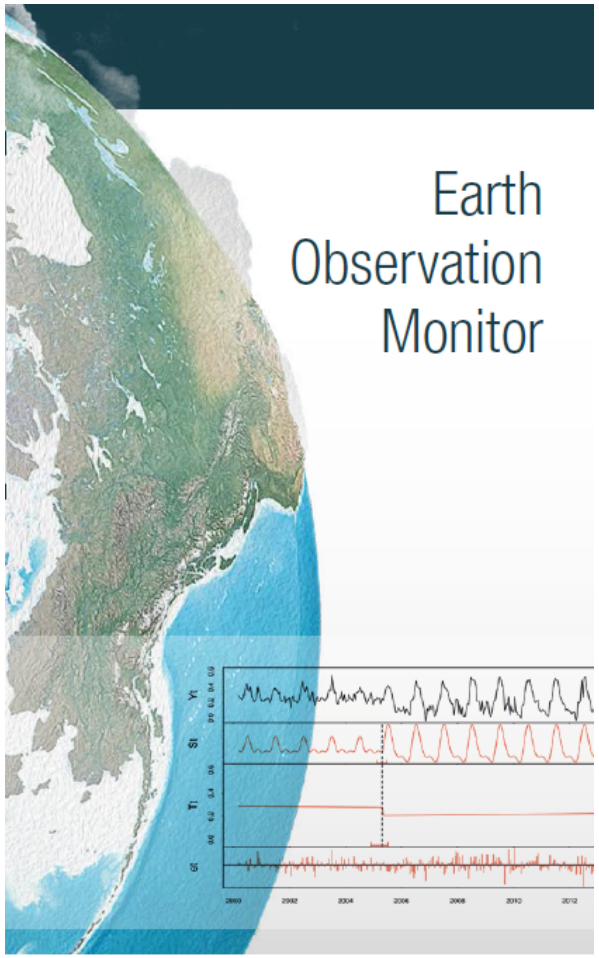
- MODIS Vegetation Produkt (MOD13Q1)
- Globale Verfügbarkeit mit der Google Earth Engine

Verfügbare WPS Dienste

- Datenintegration für Punkt und Polygone
- Trendberechnungen
- Breakpoint-Berechnungen
- Ableitung phänologischer Parameter

Client-Anwendungen

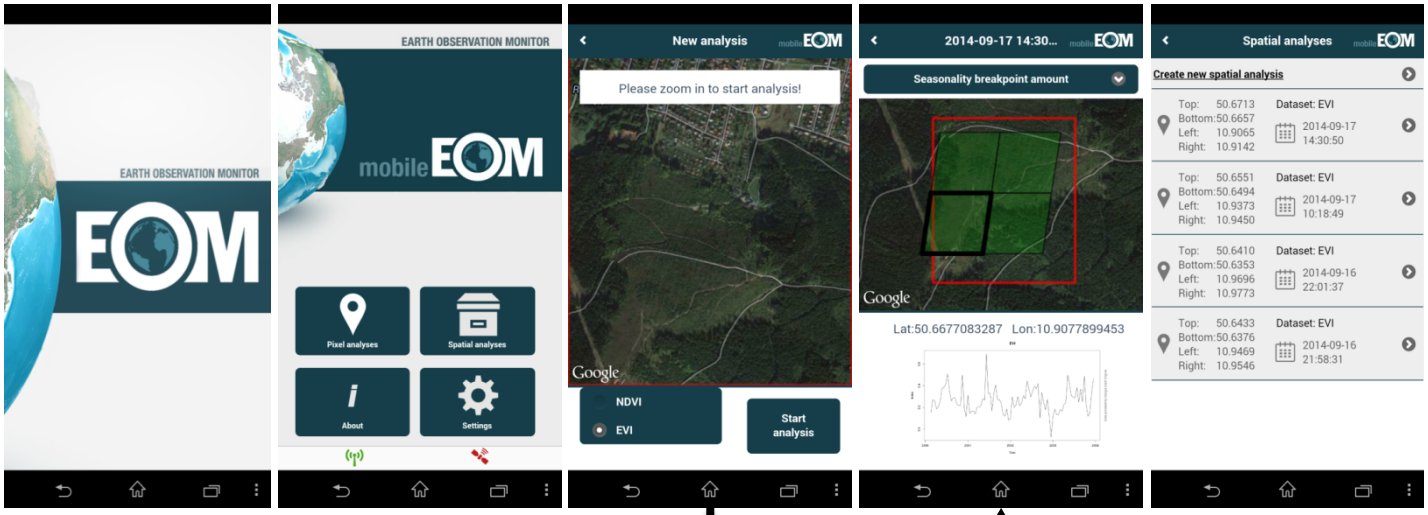
- Webportal: www.earth-observation-monitor.net
- Mobile App für Android & iOS: mobileEOM

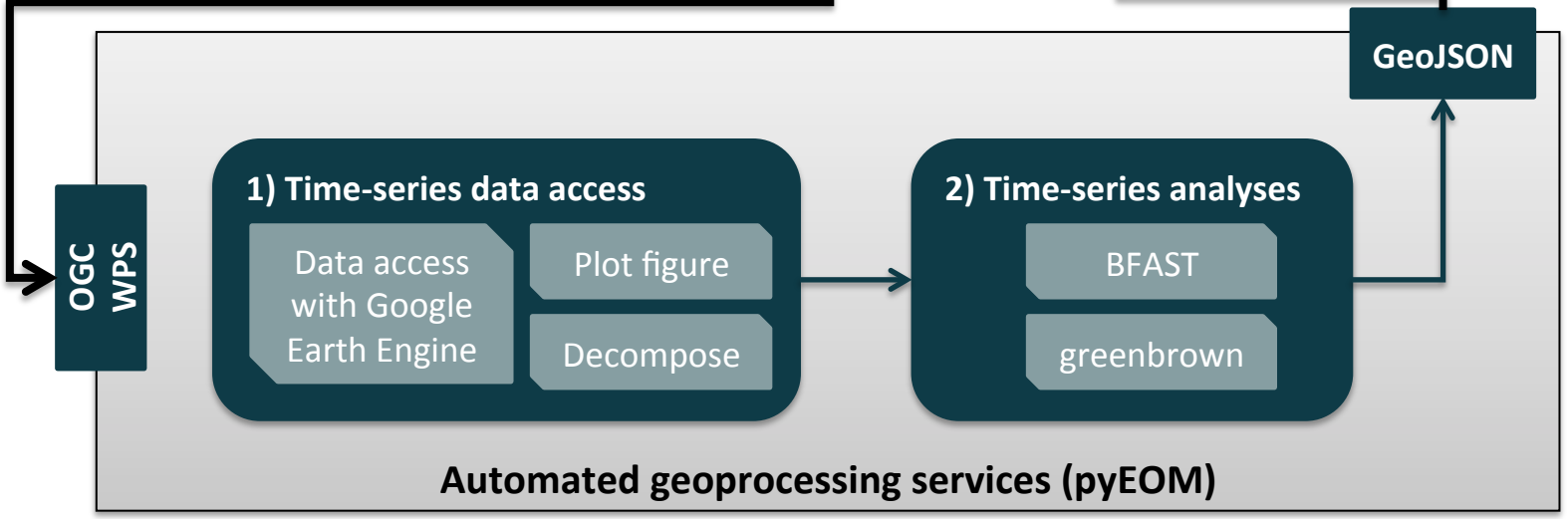


Client-Anwendung: mobileEOM

ANDROID APP ON Google play

Download on the App Store





Live-Demo

http://artemis.geogr.uni-jena.de/eoscience20/fossgis.html

Coordinate transformation

DGM-Profile with GRASS GIS

Earth Observation time-series tools

OGC Web Processing Services

Google Earth Engine

- NASA MODIS
- Landsat 1-8
- Sentinel-1

ESA Science Data Hub

- Sentinel-1

PyWPS Workshop auf der FOSS4G:

Vielen Dank für die Aufmerksamkeit.

Fragen?

Kontaktinformationen

Jonas Eberle
Friedrich-Schiller-University
Institute for Geography
Department Earth Observation
Loebdergraben 32
07743 Jena, Germany

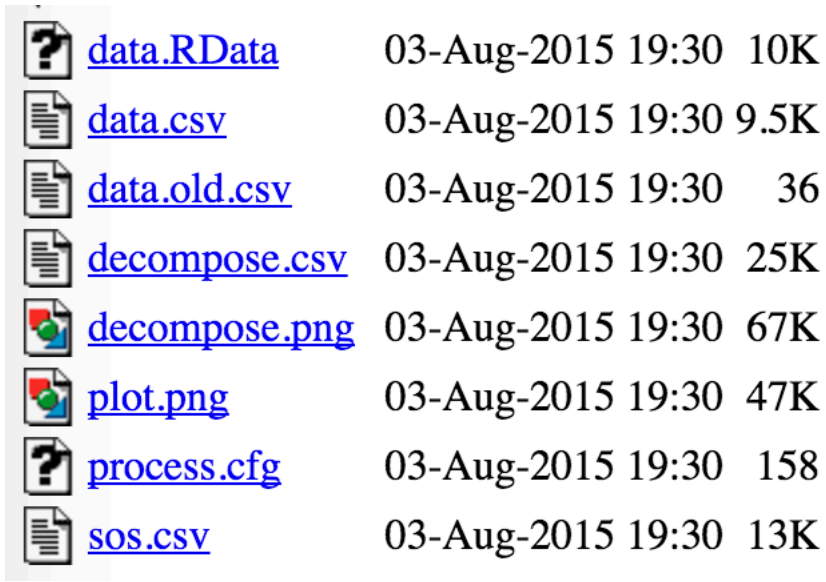
phone: +49 3641 94 88 89
email: jonas.eberle@uni-jena.de

WPS for data access

MODIS Vegetation pixel time-series

```
http://artemis.geogr.uni-jena.de/cgi-bin/testbox.cgi?service=WPS&version=
1.0.0&request=Execute&identifier=1013_single_ts_plot_point&datainputs=[dat
asetName=mod13q1_evi;pointX=13.54;pointY=52.31]
```

http://artemis.geogr.uni-jena.de/pywps/tmp/b5b17389-05d7-4058-9d87-aea05a0234d7/data/



data.RData	03-Aug-2015 19:30	10K	
data.csv	03-Aug-2015 19:30	9.5K	
data.old.csv	03-Aug-2015 19:30	36	
decompose.csv	03-Aug-2015 19:30	25K	
decompose.png	03-Aug-2015 19:30	67K	
plot.png	03-Aug-2015 19:30	47K	
process.cfg	03-Aug-2015 19:30	158	
sos.csv	03-Aug-2015 19:30	13K	

	date	value	quality	interpolated
1	2000-02-18	0.153	1	0
2	2000-03-05	0.1364	0	0
3	2000-03-21	0.1416	0	0
4	2000-04-06	0.1719	1	0
5	2000-04-22	0.1931	0	0
6	2000-05-08	0.2309	1	0
7	2000-05-24	0.2801	1	0
8	2000-06-09	0.3254	1	0
9	2000-06-25	0.3882	1	0
10	2000-07-11	0.2612	1	0
11	2000-07-27	0.281133333333333	3	1
12	2000-08-12	0.301066666666667	3	1

Output uuid: **b5b17389-05d7-4058-9d87-aea05a0234d7**

WPS for data access

MODIS Vegetation polygon time-series

```
http://artemis.geogr.uni-jena.de/cgi-bin/testbox.cgi?service=WPS&version=
1.0.0&request=Execute&identifier=1013_single_ts_plot_polygon&datainputs=[d
atasetName=mod13q1_evi;wkt=POLYGON((13.59 55.79, 14.72 55.84, 14.10 58.48,
13.00 58.43, 13.59 55.79))]
```

http://artemis.geogr.uni-jena.de/pywps/tmp/1109b91c-34d7-4404-bfe4-02d55ea4b1c0/

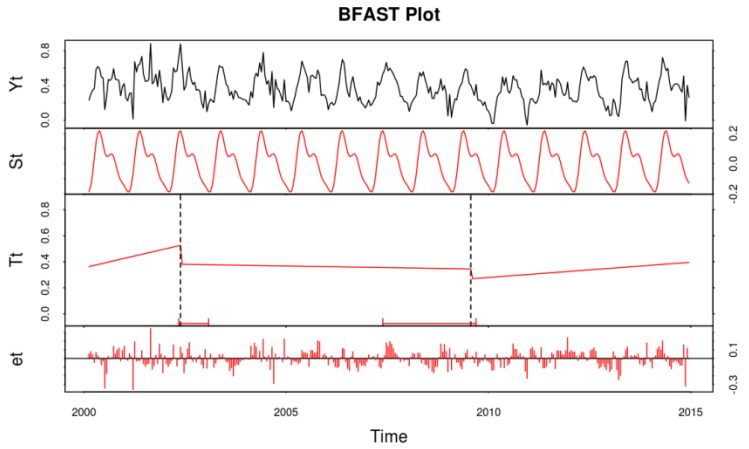
```
data/
├── data.csv
├── files.txt
├── GEE_data.json
├── output
│   ├── analysis_clipped.vrt
│   ├── MOD13Q1.A2000049.EVI.tif
│   ├── MOD13Q1.A2000065.EVI.tif
│   ├── MOD13Q1.A2000081.EVI.tif
│   ├── MOD13Q1.A2000097.EVI.tif
│   ├── ...
│   └── MOD13Q1.A2015305.EVI.tif
├── plot.png
├── polygon_modis.wkt
├── polygon_wgs84.wkt
├── process.cfg
└── ts.info
```

Polygon-based time-series observations (EVI example) in the final GeoTiff raster file format (output folder, files.txt), original files from Google Earth Engine (GEE_data.json) and further statistical summaries (data.csv, plot.png)

WPS for data analysis

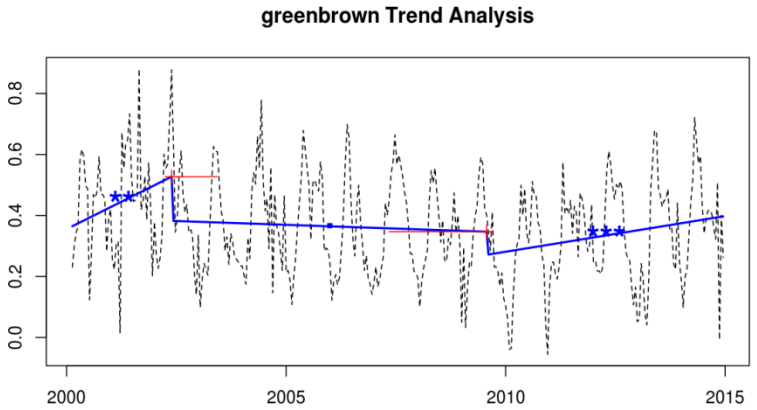
Breakpoints for vegetation time-series

```
http://artemis.geogr.uni-jena.de/cgi-bin/
testbox.cgi?
service=WPS&version=1.0.0&request=Execute&i
dentifier=2010_single_ts_bfast_point&datain
puts=[uuid=b5b17389-05d7-4058-9d87-
aea05a0234d7]
```



Trends for vegetation time-series data

```
http://artemis.geogr.uni-jena.de/cgi-bin/
testbox.cgi?
service=WPS&version=1.0.0&request=Execute&
identifier=3010_single_ts_greenbrown_point
&datainputs=[uuid=b5b17389-05d7-4058-9d87-
aea05a0234d7]
```



WPS for data discovery

Sentinel-1 ESA Data Hub – Overlapping areas

```
http://artemis.geogr.uni-jena.de/cgi-bin/testbox.cgi?service=WPS&version=
1.0.0&request=Execute&identifier=s1_datahub_test&datainputs=[wkt=POLYGON((
13.59 55.79, 14.72 55.84, 14.10 58.48, 13.00 58.43, 13.59 55.79));
product=GRD;maxoverlap=70]&rawdataoutput=output
```

Title,	Overlap,	Link,	Geometry
S1A_EW_GRDM_1SDH_20150427T…,	0.94,	Download,	POLYGON ((...))
S1A_EW_GRDM_1SDH_20150427T…,	1.00,	Download,	POLYGON ((...))
S1A_EW_GRDM_1SDH_20150422T…,	0.87,	Download,	POLYGON ((...))
S1A_EW_GRDM_1SDH_20150415T…,	0.94,	Download,	POLYGON ((...))
S1A_EW_GRDM_1SDH_20150415T…,	1.00,	Download,	POLYGON ((...))
S1A_EW_GRDM_1SDH_20150410T…,	0.87,	Download,	POLYGON ((...))
...			

WPS for data discovery

Landsat and Google Earth Engine example

```
http://artemis.geogr.uni-jena.de/cgi-bin/testbox.cgi?service=WPS&version=
1.0.0&request=Execute&identifier=gee_agu_getscenes&datainputs=[wkt=POINT(
11 51);dataset=LANDSAT/LC8_L1T;start=2013-11-01;end=2014-07-20;
maxcloudcover=30]
```

```
ID;                          Date;          Cloudcover
LC81940242013357LGN00; 2013-12-23;      5.4
LC81940242014056LGN00; 2014-02-25;      11.03
LC81940242014072LGN00; 2014-03-13;      0.2
LC81940242014136LGN00; 2014-05-16;      20.73
LC81940242014184LGN00; 2014-07-03;      1.46
LC81940242014200LGN00; 2014-07-19;      0.94
```